

**Commonwealth of Massachusetts  
Executive Office of Health and Human Services**

*August 2005*

*Version 1.0*



**Vendor Interface Specification Guide**

# Commonwealth of Massachusetts Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

## Table of Contents

1.0	Introduction.....	1
1.1.	Background.....	1
1.1.1	HIPAA Batch Transactions Processing.....	1
1.1.2	Web Interface.....	2
1.2.	Web Interface Specification.....	3
1.2.1	Connecting to the Server .....	3
1.2.2	HTTPS Messages.....	4
1.2.3	File Upload Message .....	4
1.2.4	Directory List Message.....	7
1.2.5	Download a File Message.....	14
1.2.6	General Response Message .....	18
2.0 -	Version Table.....	20
Appendix A -	Sample Upload a File Program .....	21
Appendix B –	DirectoryListing Sample Program.....	32
Appendix C -	Download a File Sample Program.....	41
Appendix D -	SSL Tunnel Through Proxy Server Utility Sample Program Sample .....	50

# Commonwealth of Massachusetts Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

## 1.0 Introduction

This document is intended for software vendors to use in order to develop applications to interact with the MassHealth Web-based system, and to upload and download HIPAA-compliant transactions in batch mode. Further information is available in the MassHealth Companion Guides and the Web-Based Transactions User Guide.



*Note: This document provides specifications for transaction exchange in batch only. The samples used in this document are coded in the Java language, but any language that supports the HTTPS protocol can be used to interact with the Web site.*

For more information please contact MassHealth Customer Service at 1-800-441-0323, or send an e-mail to providersupport@mahealth.net.

### 1.1. Background

#### 1.1.1 HIPAA Batch Transactions Processing

Proprietary claims as well as the following HIPAA-compliant transactions marked with an 'X' are available for either upload or download using the Web interface.

Function	820	834	835	837	997	EMC (proprietary)
Upload				X		
Download	X	X	X		X	X

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

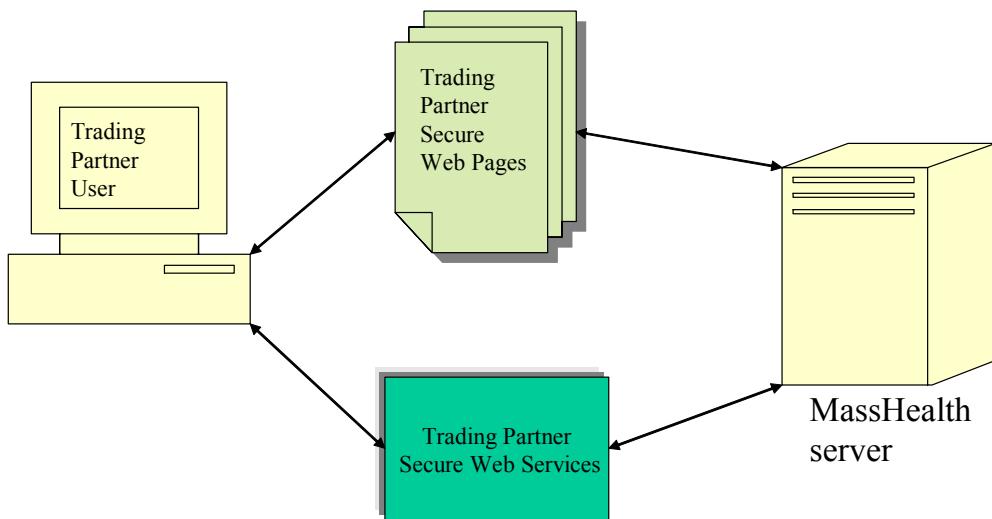
*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

### 1.1.2 Web Interface

The Web interface is designed to support batch file uploads and downloads. There are two ways to use this interface. The first way is to log on to the secure Web site using a user ID and password. This Web site has Web pages that allow users to upload and download files.

The second way is to use a software program that runs on a user's PC or on a user's server that connects to the secure submitter Web services. This site sends a request using the HTTPS protocol containing information that includes user ID, password, and the request data. The request data can include a request for a listing of files available for download, a specific file name to be downloaded, or a file to upload. The files can be transferred in compressed format or in standard ASCII text format. All data is transferred using the Secure Socket Layer (SSL) that encrypts the data over the public Internet to ensure protected health information is not exposed.



# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

### 1.1.3 Client software

Client software can be written in any language that supports HTTPS in communicating with the submitter Web services. The request transaction data is formatted in XML, but the data files transferred from and to the Web services are in the HIPAA-standard formats. The XML data is used to support the security and general interaction with the submitter Web services.

## 1.2. Web Interface Specification

### 1.2.1 Connecting to the Server

To connect to the Web application, you must first have a network connection that provides access to the public Internet or provides a dial-up connection through the Remote Access Server (RAS).

If your connection requires a proxy server to connect to the Internet, it needs to be configurable in your application (please see the sample Java program in Appendix A). Once connected, you should use the URLs in Table 1 to establish a connection with the application. It is recommended that these URLs not be hard coded as part of your application but be configurable instead. The sample program provided in Java demonstrates how to establish a connection to these URLs.

All transactions require that the connection be made using SSL. The sample program provides an example of how to do this with Java. Other programming languages such as Perl and Visual Basic have other ways to connect using SSL, for example, by using the WinInet libraries that are part of Microsoft Windows. A test environment is also available for sending test transactions.

Table 1.

Environment	Root URL
Production	<a href="https://masshealth2.ehs.state.ma.us/transactions">https://masshealth2.ehs.state.ma.us/transactions</a>
Test	<a href="https://masshealth2.ehs.state.ma.us/transactions/test">https://masshealth2.ehs.state.ma.us/transactions/test</a>

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

To access the specific Web applications, please append the URL suffixes as identified in Table 2 below.

Table 2.

Web Application	URL Suffix
File upload	/WebUploadFromClient
Directory list	/WebDirectoryDownloadFromClient
File download	/WebDownloadFromClient

### 1.2.2 HTTPS Messages

Standard messages must be sent to interact with the Web applications once a connection is established. The body of the message is formatted in simple XML. XML messages are described in detail in the following sections. A standard HTTP header must be provided including a content-type with a value of text/xml for file directory list requests and file download requests. For file uploads, it must be multipart/form-data followed by the boundary. Please see the sample requests for each transaction type for an example of the HTTP header. Most tools and programming languages that support HTTP will provide a standard header.

The message for uploading files uses the standard HTTP Multipart MIME type format. The Multipart MIME type format is the same format used to upload files with a Web browser. All other messages use HTTPS and the MIME type text. The XML data received should be parsed using a standard XML parser.

### 1.2.3 File Upload Message

The file upload message must be in the HTTPS Multipart MIME type format. There are two parts to the message. The first part is user identification data in XML format, the second is the actual file being uploaded. The file being uploaded must be in an ASCII text format or zip format. The response to this message is a general response message. Please see the “General Response Message” section for details.

When using the multipart form type, the data should be separated by a unique string of characters, which serves as the boundary between the parts of data. This number must be unique and not a part of the actual data uploaded in the request. In the multipart mime type HTTP header, the boundary characters are defined (please see sample request). The first part of the request is an XML message that includes security information to authenticate that the user has access to perform the upload (see Table 3).

# Commonwealth of Massachusetts Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

Version 1.0

A sample of the XML request and its associated schema are presented on the following pages. The data file being uploaded then follows the request. Please note that no more than one file can be uploaded at a time and the total length of the entire upload message cannot exceed 16 megabytes or the whole transaction will be rejected.

See Appendix A for sample upload program.

URL: <https://masshealth2/transactions/WebUploadFromClient> Production

URL: <https://masshealth2/transactions/test/WebUploadFromClient> Test

Table 3. File Upload XML Message Description

Element	Description	Length	Require
SubmitterId	The submitter ID is used to identify for whom the transaction is intended. The ID, along with the user ID and password, is used to authenticate the user.  Sample data: 1234567 Format: Alphanumeric	7	Y
UserId	The user ID is used to authenticate the user submitting the request. The user ID must be authorized to submit for the submitter ID. This ID is the same as the submitter ID, except when a subordinate user ID is used to submit on behalf of the submitter ID.  Sample data: 1234567 Format: Alphanumeric	7 – 15	Y
Password	The password is used in conjunction with the user ID to ensure the validity of the user making the request.  Sample data: Jmstp#567 Format: Alphanumeric	8 – 25	Y
Function	This is the upload file function name. This value must always be filled with “UPLOADFILE” to be a valid request.  Required value: UPLOADFILE Format: Alphanumeric	10	Y
FileName	The path and name of the file being uploaded.  Sample data: d:\myfile.dat Format: Alphanumeric	1 – 256	Y

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

Element	Description	Length	Require
FileSize	This specifies the size of the file. The size is compared against the actual size of the file uploaded to ensure that no data is lost. If the file is in a zipped format, the size is the zipped file size.  Sample data: 1022  Format: Numeric	8	Y

### Sample HTTP Multipart File Upload Request

```
POST /transactions/WebUploadFromClient HTTP/1.0
Content-Type: multipart/form-data; boundary=7d021a37605f0
User-Agent: Java1.2.2
Host: masshealth2.ehs.state.ma.us
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Content-length: 698

--7d021a37605f0
Content-Disposition: form-data; name="message"

<?xml version="1.0" encoding="UTF-8"?>
<UploadRequest>
  <IdentificationHeader>
    <SubmitterId>1234567</SubmitterId>
    <UserId>1234567</UserId>
    <Password>Jmstp#567</Password>
  </IdentificationHeader>
  <Transaction>
    <Function>uploadFile</Function>
    <FileName>d:\myfile.dat</FileName>
    <FileSize>45</FileSize>
  </Transaction>
</UploadRequest>
--7d021a37605f0
Content-Disposition: form-data;name="textFileAttached"; filename="d:\temp\test.txt"
Content-Type: text/plain

File in X12N format to upload.
X12N data
X12N data
--7d021a37605f0
```

# Commonwealth of Massachusetts Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

Version 1.0

## File Upload Request XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="FileName" type="xsd:string"/>
    <xsd:element name="FileSize" type="xsd:string"/>
    <xsd:element name="Function" type="xsd:string"/>
    <xsd:element name="IdentificationHeader">
        <xsd:complexType>
            <xsd:sequence minOccurs="1" maxOccurs="1">
                <xsd:element ref="SubmitterId" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="UserId" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="Password" minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="Password" type="xsd:string"/>
    <xsd:element name="SubmitterId" type="xsd:string"/>
    <xsd:element name="Transaction">
        <xsd:complexType>
            <xsd:sequence minOccurs="1" maxOccurs="1">
                <xsd:element ref="Function" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="FileName" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="FileSize" minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="UploadRequest">
        <xsd:complexType>
            <xsd:sequence minOccurs="1" maxOccurs="1">
                <xsd:element ref="IdentificationHeader" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="Transaction" minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="UserId" type="xsd:string"/>
</xsd:schema>
```

### 1.2.4 Directory List Message

The directory list message is an XML formatted message sent over HTTPS. The reason for a directory list is to get the name of any available files for downloading. Each file name must be specified in the download message in order to get the file. There are several options that can be specified in the XML message to filter what is returned in the directory list (see Table 4).

There are two possible return type messages. If the directory listing is successful, a directory listing response message is returned (see Table 5). The directory listing response message contains all the files meeting the criteria specified in the request message. If the directory listing request message is incorrectly formatted, or there are problems processing the request, a general response message is returned with a description of the error. Please see the “[General Response Message](#)” section for details.

# Commonwealth of Massachusetts Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

Version 1.0

See Appendix B for directory list sample program.

URL: <https://masshealth2/transactions/WebUploadFromClient> Production

URL: <https://masshealth2/transactions/test/WebUploadFromClient> Test

Table 4. Directory Listing Request Message Description

Element	Description	Length	Require
SubmitterId	<p>The submitter ID is used to identify for whom the transaction is intended. The ID is used to limit the list of files in the directory list to only those belonging to this submitter. The submitter ID is cross-validated against the user ID, and the user ID must be authorized to submit requests for this submitter ID.</p> <p>Sample data: 1234567</p> <p>Format: Alphanumeric</p>	7	Y
UserId	<p>The user ID is used to authenticate the user submitting the request. The user ID must be authorized to submit for the submitter ID. This ID is the same as the submitter ID, except when a subordinate user ID is used to submit on behalf of the submitter ID.</p> <p>Sample data: 1234567</p> <p>Format: Alphanumeric</p>	7-15	Y
Password	<p>The password is used in conjunction with the user ID to ensure the validity of the user making the request.</p> <p>Sample data: Jmstp#567</p> <p>Format: Alphanumeric</p>	8 – 25	Y
Function	<p>This is the directory list function name. This value must always be filled with “DIRLIST” to be a valid request.</p> <p>Required value: DIRLIST</p> <p>Format: Alphanumeric</p>	7	Y

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

Element	Description	Length	Require
FilesToReturnCount	This specifies the maximum number of files to return in the directory list. If not included, then all files are returned.  Sample data: 50  Format: Numeric	0-6	N
FileType	This indicates the types of files you want returned in the directory list. For example, if you want only the 997 functional acknowledgments, you would specify a file type of 997. If no file type is specified, then all file types are returned.  820 – Health plan payment 834 – Benefit enrollment 835 – Remittance advice 997 – Functional acknowledgment EMC – Supplemental electronic emittance Advice	3	N
ProviderDirectory	This element is optional and by default; if not supplied, the directory listing returned will be for the submitter's ID. If the submitter has been granted access to other provider directories, the provider ID can be supplied with the request to return that provider's directory of files.  Sample data: 7777777  Format: Alphanumeric	7-15	N
FileStatus	Indicator to select files based on the following: A = All files N = New files only D = Previously downloaded files only	1	Y

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

Element	Description	Length	Require
FilesCreatedFromDate	<p>This is used as a filter to return only files in the directory list greater than or equal to this date. If this element is not specified, all files are listed, up through the FileCreatedToDate, if specified.</p> <p>Sample data: 20020701</p> <p>Format: 4-digit year, 2-digit month, 2-digit day</p>	8	N
FilesCreatedToDate	<p>This is used as a filter to return only files in the directory list less than or equal to this date. If this element is not specified, all files are listed, equal to and greater than the FileCreatedFromDate, if specified. If the from or to date element is specified, all files are returned in the directory list.</p> <p>Sample data: 20020731</p> <p>Format: 4-digit year, 2-digit month, 2-digit day</p>	8	N

### Sample HTTP Request for Directory Listing

```

POST /transactions/WebDirectoryDownloadFromClient HTTP/1.0
Content-Type: text/xml
User-Agent: Java1.2.2
Host: masshealth2.ehs.state.ma.us
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Content-length: 708

<?xml version="1.0" encoding="UTF-8"?>
<DirectoryRequest>
  <IdentificationHeader>
    <SubmitterId>1234567</SubmitterId>
    <UserId>1234567</UserId>
    <Password>Jmstp#567</Password>
  </IdentificationHeader>
  <Transaction>
    <Function>DIRLIST</Function>
    <FilesToReturnCount>50</FilesToReturnCount>
    <SelectedFileTypes>
      <FileType>997</FileType>
    </SelectedFileTypes>
    <ProviderDirectory>7777777</ProviderDirectory>
    <FileStatus>A</FileStatus>
    <FilesCreatedFromDate>20020102</FilesCreatedFromDate>
    <FilesCreatedToDate>20020103</FilesCreatedToDate>
  </Transaction>
</DirectoryRequest>

```

### Directory Listing Request XML Schema

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" version="1.0" xml:lang="EN">
  <xsd:element name="DirectoryRequest">
    <xsd :complexType>
      <xsd :sequence minOccurs= »1 » maxOccurs= »1 »>
        <xsd :element ref= »IdentificationHeader » minOccurs= »1 » maxOccurs= »1 »/>

```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

```
<xsd :element ref= »Transaction » minOccurs= »1 » maxOccurs= »1 »/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>

<xsd:element name="SubmitterId" type="xsd:string"/>
<xsd:element name="UserId" type="xsd:string"/>
<xsd:element name="Password" type="xsd:string"/>

<xsd:element name="IdentificationHeader">
  <xsd:complexType>
    <xsd:sequence minOccurs="1" maxOccurs="1">
      <xsd:element ref="SubmitterId" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="UserId" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="Password" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="Function" type="xsd:string"/>
<xsd:element name="FilesToReturnCount" type="xsd:string"/>
<xsd:element name="FileType" type="xsd:string"/>
<xsd:element name="SelectedFileTypes">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="FileType" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="ProviderDirectory" type="xsd:string"/>
<xsd:element name="FilesStatus" type="xsd:string"/>
<xsd:element name="FilesCreatedFromDate" type="xsd:string"/>
<xsd:element name="FilesCreatedToDate" type="xsd:string"/>

<xsd:element name="Transaction">
  <xsd:complexType>
    <xsd:sequence minOccurs="1" maxOccurs="1">
      <xsd:element ref="Function" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="FilesToReturnCount" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="SelectedFileTypes" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="ProviderDirectory" minOccurs="0" maxOccurs="1"/>
      <xsd:element ref="FilesStatus" minOccurs="0" maxOccurs="1"/>
      <xsd:element ref="FilesCreatedFromDate" minOccurs="0" maxOccurs="1"/>
      <xsd:element ref="FilesCreatedToDate" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
</xsd:schema>
```

Table 5. Directory Listing Response Message Description

Element	Description	Length	Require
SubmitterId	The submitter ID from the request.  Sample data: 1234567  Format: Alphanumeric	7	Y

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

Element	Description	Length	Require
UserId	The user ID from the request. This ID is the same as the submitter ID, except when a Subordinate User ID is used to submit on behalf of the submitter ID.  Sample data: 1234567 Format: Alphanumeric	7-15	Y
Function	The function from the request.  Required value: DIRRESP Format: Alphanumeric	7	Y
FilesReturned Count	The number of files returned in the directory listing.  Sample data: 50 Format: Numeric	1 – 6	Y
FileName	The file name given to the file. This is a system-generated name. This is the same name needed to request the file for download.  Sample data: 1234567_20050419121525_00000000_835_R1 234.dat Format: Alphanumeric	39-60	Y
FileCreationDate	The date the file was created.  Sample data: 20020701 Format: 4-digit year, 2-digit month, 2-digit day	8	Y
FileType	This is the type of file.  Sample data: 835	3	Y
FileSize	The size in bytes of the file in unzipped format.  Sample data: 50000 Format: Numeric	1 – 8	Y
LastDownload Date	The date the file was last downloaded. If the file has not been downloaded, this is empty or blank.  Sample data: 20020701 Format: 4-digit year, 2-digit month, 2-digit day.	8	N

Sample HTTP Response from Directory Listing Request

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

```
HTTP/1.1 200 OK
Date: Mon, 05 Aug 2002 17:39:13 GMT
Server: IBM_HTTP_Server/1.3.12.3 Apache/1.3.12 (Win32)
Last-Modified: Fri, 26 Jul 2002 21:24:30 GMT
ETag: "0-2dae-3d41be0e"
Accept-Ranges: bytes
Content-Length: 11694
Connection: close
Content-Type: text/xml

<?xml version="1.0" encoding="UTF-8"?>
<DirectoryResponse>
    <IdentificationHeader>
        <SubmitterId>1234567</SubmitterId>
        <UserId>1234567</UserId>
    </IdentificationHeader>
    <Transaction>
        <Function>DIRLIST</Function>
        <FilesReturnedCount>2</FilesReturnedCount>
        <DirectoryList>
            <FileName>1234567_20050419121525_00000000_835_R1234.dat </FileName>
            <FileCreationDate>20050419</FileCreationDate>
            <FileType>835</FileType>
            <FileSize>1210</FileSize>
            <LastDownLoadDate></LastDownLoadDate>
        </DirectoryList>
        <DirectoryList>
            <FileName>1234567_20050406122035_20050407_835_R1235.dat </FileName>
            <FileCreationDate>20050406</FileCreationDate>
            <FileType>835</FileType>
            <FileSize>120</FileSize>
            <LastDownLoadDate>20050407</LastDownLoadDate>
        </DirectoryList>
    </Transaction>
</DirectoryResponse>
```

### Directory Listing Response XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="DirectoryList">
        <xsd:complexType>
            <xsd:sequence minOccurs="0" maxOccurs="450">
                <xsd:element ref="FileName" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="FileCreationDate" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="FileType" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="FileSize" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="LastDownLoadDate" minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="DirectoryResponse">
        <xsd:complexType>
            <xsd:sequence minOccurs="1" maxOccurs="1">
                <xsd:element ref="IdentificationHeader" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="Transaction" minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="FileCreationDate" type="xsd:string"/>
    <xsd:element name="FileName" type="xsd:string"/>
    <xsd:element name="FileSize" type="xsd:string"/>
    <xsd:element name="LastDownLoadDate" type="xsd:string"/>
    <xsd:element name="FileType" type="xsd:string"/>
    <xsd:element name="FilesReturnedCount" type="xsd:string"/>
    <xsd:element name="Function" type="xsd:string"/>
    <xsd:element name="IdentificationHeader">
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

Version 1.0

```
<xsd:complexType>
    <xsd:sequence minOccurs="1" maxOccurs="1">
        <xsd:element ref="SubmitterId" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="UserId" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>
</xsd:element>

<xsd:element name="SubmitterId" type="xsd:string"/>
<xsd:element name="Transaction">
    <xsd:complexType>
        <xsd:sequence minOccurs="1" maxOccurs="1">
            <xsd:element ref="Function" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="FilesReturnedCount" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="DirectoryList" minOccurs="1" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="UserId" type="xsd:string"/>
</xsd:schema>
```

### 1.2.5 Download a File Message

The download a file message is an XML-formatted message sent over HTTPS. A file name from the directory listing response is used to request the file to download. This file name must be specified in the download message in order to get the file. The request is made using XML (see Table 6 for details).

There are two possible returns from a download request. If the file requested was found and there were no problems processing the request, the data file is returned. If any problems occurred processing the request, a general response message is returned. The client application must recognize the difference between a file being returned and the general response message. Please see the “[General Response Message](#)” section for details.

See Appendix C for download sample program.

URL: <https://masshealth2/transactions/WebUploadFromClient> Production

URL: <https://masshealth2/transactions/test/WebUploadFromClient> Test

Table 6. Download a File Message Description

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

Element	Description	Length	Require
SubmitterId	<p>The submitter ID is used to identify for whom the transaction is intended. The submitter ID is cross-validated against the user ID, and the user ID must be authorized to submit requests for this submitter ID.</p> <p>Sample data: 1234567</p> <p>Format: Alphanumeric</p>	7	Y
UserId	<p>The user ID is used to authenticate the user submitting the request. The user ID must be authorized to submit for the submitter ID.</p> <p>Sample data: 1234567</p> <p>Format: Alphanumeric</p>	7-15	Y
Password	<p>The password is used in conjunction with the user ID to ensure the validity of the user making the request.</p> <p>Sample data: Jmstp#567</p> <p>Format: Alphanumeric</p>	8-25	Y
Function	<p>This is the file download function name. This value must always be filled with "DOWNLOAD" to be a valid request.</p> <p>Required value: DOWNLOAD</p> <p>Format: Alphanumeric</p>	8	Y
ProviderDirectory	<p>This element is optional and by default; if not supplied, files will be downloaded from the submitter's directory. If the submitter has been granted access to other provider directories and wants to download a file from one of these provider directories, the ProviderDirectory element must be specified. In other words, if the FileName to be downloaded does not exist in the submitter's default directory, a ProviderDirectory element must be specified to find the file.</p> <p>Sample data: 7777777</p> <p>Format: Alphanumeric</p>	7-15	N
FileName	<p>This specifies the name of the file to download. The name can be found from the directory list request.</p> <p>Sample data: 7777777_20050419121525_0000000_835_R1234.dat</p> <p>Format: Alphanumeric</p>	39 - 60	N

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

Element	Description	Length	Require
FileFormat	<p>This specifies the file format you want the file downloaded in.</p> <p>Valid values are:</p> <p>TXT – Requests the file in standard ASCII Text format.</p> <p>ZIP – Requests the file in zipped format.</p>	3	Y

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

**MassHealth Interface Specification Guide**  
August 2005

**Version 1.0**

### Sample HTTP Request for a File Download

```
POST /transactions/WebDownloadFromClient HTTP/1.0
Content-Type: text/xml
User-Agent: Java1.2.2
Host: masshealth2.ehs.state.ma.us
Accept: text/html, image/gif, image/jpeg, *, *; q=.2, */*; q=.2
Content-length: 359
<?xml version="1.0" encoding="UTF-8"?>
<DownloadRequest>
    <IdentificationHeader>
        <SubmitterId>1234567</SubmitterId>
        <UserId>1234567</UserId>
        <Password>Jmstp#567</Password>
    </IdentificationHeader>
    <Transaction>
        <Function>DOWNLOAD</Function>
        <ProviderDirectory>7777777</ProviderDirectory>
        <FileName>7777777_20050419121525_00000000_835_R1234.dat </FileName>
        <FileFormat>TXT</FileFormat>
    </Transaction>
</DownloadRequest>
```

### Download File Request XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="DownloadRequest">
        <xsd:complexType>
            <xsd:sequence minOccurs="1" maxOccurs="1">
                <xsd:element ref="IdentificationHeader" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="Transaction" minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="FileName" type="xsd:string"/>
    <xsd:element name="FileFormat" type="xsd:string"/>
    <xsd:element name="Function" type="xsd:string"/>
    <xsd:element name="ProviderDirectory" type="xsd:string"/>
    <xsd:element name="IdentificationHeader">
        <xsd:complexType>
            <xsd:sequence minOccurs="1" maxOccurs="1">
                <xsd:element ref="SubmitterId" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="UserId" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="Password" minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="Password" type="xsd:string"/>
    <xsd:element name="SubmitterId" type="xsd:string"/>
    <xsd:element name="Transaction">
        <xsd:complexType>
            <xsd:sequence minOccurs="1" maxOccurs="1">
                <xsd:element ref="Function" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="ProviderDirectory" minOccurs="0" maxOccurs="1"/>
                <xsd:element ref="FileName" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="FileFormat" minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="UserId" type="xsd:string"/>
</xsd:schema>
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

### 1.2.6 General Response Message

The general response message can be returned for any type request message. This response message indicates whether a request was successful or not. Some requests, such as the directory request, do not return a general response message. Instead a directory listing response message is returned. If, however, there is a problem returning the directory listing response, a general response message will be returned instead.

#### File Upload Response Message Description

Element	Description	Length	Require
MessageCode	<p>Code identifying the message. Sample data: 100 Format: Numeric</p> <p>The codes can range from 0 – 999. The message code of 0 is the only successful code. The code 999 is a generic system error and other error codes may exist. These codes may change, so it is recommended that you not check for any specific error codes and but rather check only for the successful code.</p> <p>0 – Successful 202 – User ID or password invalid 203 – Password expired 204 – User ID suspended 205 – User not authorized to perform this function 207 – Upload file is zero bytes 208 – No file selected to upload 211 – Submitter mismatch with user ID 212 – Invalid function code 213 – Provider ID mismatch for submitter 214 – Requested file not available for download 215 – I/O error zipping file for download 216 – Filename in XML doesn't match HTTP file name 217 – File size in XML does not match file length 999 – Unable to process request</p>	1 - 3	Y

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

Element	Description	Length	Require
MessageDesc	Descriptive message.  Sample data: File was successfully uploaded.  Format: Alphanumeric	Up to 256	Y
MessageType	Code identifying the type of message this is.  Sample data: A.  Format: Alpha  Codes: A – Transaction accepted with no errors.  W – Transaction failed, please see message for more information and retry.  E – Currently the system is unavailable.	1	Y
ControlNumber	This is the transaction control number. This is currently used only by upload transactions responses.  Format: Numeric	1 – 18	N

### Sample HTTP Response Message

```

HTTP/1.1 200 OK
Date: Mon, 05 Aug 2002 17:39:13 GMT
Server: IBM HTTP Server/1.3.12.3 Apache/1.3.12 (Win32)
Last-Modified: Fri, 26 Jul 2002 21:24:30 GMT
ETag: "0-2dae-3d41be0e"
Accept-Ranges: bytes
Content-Length: 11694
Connection: close
Content-Type: text/xml

<?xml version="1.0" encoding="UTF-8"?>
<ResponseMessage>
  <Messages>
    <MessageCode>0</MessageCode>
    <MessageDesc>Successful Upload</MessageDesc>
    <MessageType>A</MessageType>
    <ControlNumber>12345</ControlNumber>
  </Messages>
</ResponseMessage>

```

# Commonwealth of Massachusetts Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

## Response Message XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="MessageCode" type="xsd:string"/>
    <xsd:element name="MessageDesc" type="xsd:string"/>
    <xsd:element name="MessageType" type="xsd:string"/>
    <xsd:element name="ControlNumber" type="xsd:string"/>
    <xsd:element name="Messages">
        <xsd:complexType>
            <xsd:sequence minOccurs="1" maxOccurs="1">
                <xsd:element ref="MessageCode" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="MessageDesc" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="MessageType" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="ControlNumber" minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="ResponseMessage">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="Messages" minOccurs="1" maxOccurs="10"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

## 2.0 - Version Table

Version	Date	Section/Pages	Description
1.0	08/01/05	<ul style="list-style-type: none"><li>• Corrected references to trading partner ID to refer to submitter ID.</li><li>• Added new ProviderDirectory element to both directory download and file download requests, and made other clarifications in the document.</li><li>• Final format and editing of entire document.</li></ul>	Production version issued

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

MassHealth Interface Specification Guide  
August 2005

Version 1.0

### Appendix A - Sample Upload a File Program

#### Upload a file program example

This sample program is presented as a guide only. It is not to be used in production without testing and possible changes. A different version of this sample may be released in response to changes that may occur.



***Important Note: The sample below and other samples in this document are coded in the Java language, but any language that supports the HTTPS protocol can be used to interact with the Web site.***

```
package com.eds.ne.regional.samples;

import com.sun.net.ssl.HttpsURLConnection;
import com.sun.net.ssl.HostnameVerifier;
import java.net.*;
import java.io.*;

/**
 * This program is provided as sample only as is without warranty or
 * condition of any kind, either expressed or implied, including, but
 * not limited to, the implied warranties of merchantability and
 * fitness for a particular purpose.
 *
 * This sample File Upload program connects to the web server then
 * sends it a HTTP Multi-Part message containing an XML message and
 * the file data to be uploaded. The name of the file to be uploaded
 * is passed to the program, along with the server name. See the
 * acceptArguments method for parameters to pass on the command line.
 * If no parameters are passed the program will prompt the user for
 * the input parameters then execute. The upload response is simply
 * displayed in standard out. This program differs from the other
 * sample programs as it builds the XML message in the code rather
 * than reading it from a file.
 *
 * See the vendor specifications for details of the format of the XML message
 * for a file upload.
 *
 * <P>
 * In order to negotiate the SSL certificate of the web server the client has
 * to have a trusted certificate in its Java keystore file. The web sites use
 * GeoTrust Certificates. If this Certificate is not in your Java installations
 * keystore you will need to install it. To install the certificate you can use
 * the java keytool command
 *
 * This sample program was written using JDK 1.2.2. In order for the sample program
 * to communicate over SSL the Java Secure Socket Extension (JSSE) needs to be installed.
 * See http://java.sun.com/products/jsse/INSTALL.html.
 * After installing JSSE add the following jar files must be in the class path before
 * running the sample program: drive:/JSSE/jsse.jar;drive:/JSSE/jnet.jar;drive:/JSSE/jcert.jar
 * <P>
 * <HR>
 * <P>
 * <TABLE BORDER="1" CELLPADDING="3" CELLSPACING="0" WIDTH="100%">
 * <TR BGCOLOR="#9393ff">
 *   <TD COLSPAN=4><FONT SIZE="+2"><B>Modification History</B></FONT></TD>
 * </TR>
 * <TR BGCOLOR="white" CLASS="TableRowColor">
```

# Commonwealth of Massachusetts Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

Version 1.0

```
* <TH>Reason</TH><TH>Date</TH><TH>Systems Engineer</TH><TH>Description of Modification</TH>*
</TR>
* <TR>
* <TD ALIGN=CENTR>EDS<?TD>
* <TD ALIGN= CENTER>NE Regional HIPAA Translator Implementation</TD>
* <TD ALIGN= CENTER>12/19/2002</TD>

* <TD>Initial deployment of this class.</TD>
* </TR>
* </TABLE>
*
*   @author Michael Smith
*
*   @see SSLTunnelSocketFactory
*/
public class ClientFileUpload
{
/**
 * Users file Upload User Id. This user id is authenticated
 * on the server before uploading the users data.
 */
    String userId = "";
/**
 * Upload User Password.
 */
    String password = "";
/**
 * Upload File drive:\path\name.
 */
    String fileName = "d:\\temp\\test.txt";
/**
 * Upload File size.
 */
    String fileSize = "0";
/**
 * Users proxy servers name or IP address.
 */
    String proxyServer = null;
/**
 * Users Proxys Port assignment.
 */
    String proxyPort = "80";
/**
 * True if Users sets their proxy server.
 */
    boolean thereIsProxy = false;
/**
 * Users Proxy Server User Id.
 */
    String proxyUser = null;
/**
 * Users proxy server password.
 */
    String proxyPassword = null;
/**
 * URL to post the data to.
 */
    String postUrl =
"https://www.ctmedicalprogram/test/secure/WebDirectoryDownloadFromClient";
/**
 * HTTP Protocol Switch used to turn on HTTP only uploads.
 */
    boolean httpSwitch = false;

    final static String XML_START = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>";
    final static String XML_UPLOAD_MESSAGE_START = "<UploadRequest>";
    final static String XML_UPLOAD_MESSAGE_END = "</UploadRequest>";
    final static String XML_IDENTIFICATION_START = "<IdentificationHeader>";
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

Version 1.0

```
final static String XML_IDENTIFICATION_END = "</IdentificationHeader>";
final static String XML_SUBMITTER_ID_START = "<SubmitterId>";
final static String XML_SUBMITTER_ID_END = "</SubmitterId>";
final static String XML_USER_ID_START = "<UserId>";
final static String XML_USER_ID_END = "</UserId>";
final static String XML_PASSWORD_START = "<Password>";
final static String XML_PASSWORD_END = "</Password>";

final static String XML_TRANSACTION_START = "<Transaction>";
final static String XML_TRANSACTION_END = "</Transaction>";
final static String XML_FUNCTION_START = "<Function>";
final static String XML_FUNCTION_END = "</Function>";
final static String XML_FILE_NAME_START = "<FileName>";
final static String XML_FILE_NAME_END = "</FileName>";
final static String XML_FILE_SIZE_START = "<FileSize>";
final static String XML_FILE_SIZE_END = "</FileSize>";

/**
 * ClientFileUpload default constructor calls it's super, then
 * setups System properties for SSL protocol.
 */
public ClientFileUpload()
{
    super();

    System.setProperty(
        "java.protocol.handler.pkgs",
        "com.sun.net.ssl.internal.www.protocol");

    // Required for ssl support and avoiding error:
    // "unknown protocol: https" exception
    java.security.Security.addProvider(new com.sun.net.ssl.internal.ssl.Provider());
}

/**
 * Arguments passed on the command line are validated and
 * set by this method.  The parameters are as follows:
 * <pre>
 * UserID Password FileName [URL] [proxyServer ProxyUser ProxyPassword]
 * </pre>
 *
 * @exception Exception - Invalid arguments message.
 */
public void acceptArgs(String[] args)
    throws Exception
{
    if (args.length >= 3)
    {
        userId        = args[0];
        password      = args[1];
        fileName      = args[2];
    }

    setSize();

    if (args.length == 4)
    {
        postUrl       = args[3];
    }

    if (args.length == 6)
    {
        proxyServer   = args[3];
        proxyUser     = args[4];
        proxyPassword = args[5];
        thereIsProxy = true;
    }

    if (args.length == 7)
    {
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

Version 1.0

```
        postUrl          = args[3];
        proxyServer      = args[4];
        proxyUser        = args[5];
        proxyPassword    = args[6];
        thereIsProxy     = true;
    }

    if (args.length == 3 || args.length == 4 || args.length == 6 || args.length == 7)
    {
        // continue
    }
    else
    {
        throw new Exception("Invalid number of parameters found:" + args.length + " must
be 3, 4, 6, or 7 \n" +
                            "Valid Params are: UserID Password FileName [URL] [proxyServer
ProxyUser ProxyPassword]");
    }
}

/**
 * Build the header information needed for the multi-part format before actually
 * sending the file data. Supports both zipped files and text file formats.
 *
 * @param streamSend StringBuffer
 */
public void buildUploadFileHeader(StringBuffer streamSend)
{
    // Each file is also sent within a boundary delimiter

    if (fileName.endsWith(".zip"))
    {
        // Zipped file format
        streamSend
            .append("Content-Disposition: form-data;name=\"zipFileAttached\"; filename=\"\"")
            .append(fileName)
            .append("\r\n")
            .append("Content-Type: application/x-zip-compressed\r\n")
            .append("\r\n");
    }
    else
    {
        // Or Plain Text File
        streamSend
            .append("Content-Disposition: form-data;name=\"textFileAttached\"; filename=\"\"")
            .append(fileName)
            .append("\r\n");
        streamSend.append("Content-Type: text/plain\r\n");
        streamSend.append("\r\n");
    }
}
/**
 * Builds the necessary format to pass parameter values with the upload file.
 *
 * @param streamSend StringBuffer
 * @param paramName java.lang.String
 * @param paramValue java.lang.String
 * @param boundry java.lang.String
 */
public void buildUploadParam(StringBuffer streamSend, String paramName, String paramValue, String
boundry)
{
    streamSend
        .append("Content-Disposition: form-data; name=\"\"")
        .append(paramName)
        .append("\r\n\r\n")
        .append(paramValue)
        .append("\r\n")
        .append("--" + boundry + "\r\n");
}
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

Version 1.0

```
}

/**
 * Builds the XML required in the Upload Message.
 *
 * @return java.lang.String
 */
public String createXmlMessage()
{
    StringBuffer xmlMessage = new StringBuffer()
        .append(XML_START)
        .append(XML_UPLOAD_MESSAGE_START)

        .append(XML_IDENTIFICATION_START)
        .append(XML_SUBMITTER_ID_START)
        .append(userId)
        .append(XML_SUBMITTER_ID_END)
        .append(XML_USER_ID_START)
        .append(userId)
        .append(XML_USER_ID_END)
        .append(XML_PASSWORD_START)
        .append(password)
        .append(XML_PASSWORD_END)
        .append(XML_IDENTIFICATION_END)

        .append(XML_TRANSACTION_START)
        .append(XML_FUNCTION_START)
        .append("uploadFile")
        .append(XML_FUNCTION_END)
        .append(XML_FILE_NAME_START)
        .append(fileName)
        .append(XML_FILE_NAME_END)
        .append(XML_FILE_SIZE_START)
        .append(fileSize)
        .append(XML_FILE_SIZE_END)
        .append(XML_TRANSACTION_END)
        .append(XML_UPLOAD_MESSAGE_END);

    return xmlMessage.toString();
}
/**
 * Main creates an instance of ClientFileUpload then
 * calls it's runProcess method.
 *
 * @param args java.lang.String[]
 */
public static void main(String[] args)
{
    ClientFileUpload clientFileUpload = new ClientFileUpload();
    clientFileUpload.setHTTP();
    clientFileUpload.runProcess(args);
}

/**
 * Override Hostname Verifier to ignore Certificate errors where the
 * URL on the certificate doesn't match the URL being accessed.
 * This should only be done for testing.
 *
 * @param theURLConnection URLConnection
 */
public void overrideHostNameVerifier(URLConnection theURLConnection)
{
    System.out.println("overrideHostNameVerifier - start");

    ((HttpsURLConnection) theURLConnection).setHostnameVerifier(new HostnameVerifier()
    {

```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

```
public boolean verify(String urlHost, String certHost)
{
    if (!urlHost.equals(certHost))
    {
        System.out.println(
            "certificate <" +
            certHost +
            "> does not match host <" +
            urlHost +
            "> but " +
            "continuing anyway");
    }
    return true;
});
}
/**
 * Post MultiPart mime data to a web server.
 *
 * @exception MalformedURLException - thrown if URL invalid.
 * @exception IOException - thrown for any IO handling errors.
 */
public void postToWeb() throws MalformedURLException, IOException
{
    // Create URL to post to.
    URL destUrl = new URL(postUrl);

    // Prepare the connection for receiving the form
    URLConnection theURLConnection = destUrl.openConnection();

    //Setup HTTPS IP Tunneling through Proxy if needed.
    if (thereIsProxy)
    {
        setProxy(theURLConnection);
    }

    // Set URL options
    theURLConnection.setDoOutput(true);
    theURLConnection.setDoInput(true);
    theURLConnection.setUseCaches(false);

    // Override Certificate error for HTTPS
    if (httpSwitch)
    {
        // continue
    }
    else
    {
        overrideHostNameVerifier(theURLConnection);
    }

    // Multi-Part delimiter (may want to randomly generate this)
    String boundary = "7d021a37605f0";

    // Set the Multi-Part Header
    theURLConnection.setRequestProperty(
        "Content-Type",
        "multipart/form-data; boundary=" + boundary);

    System.out.println("Before DataOutputStream");

    DataOutputStream postFormFile =
        new DataOutputStream(theURLConnection.getOutputStream());

    // Buffer used to build stream of multi-part form field
    StringBuffer streamSend = new StringBuffer();

    // Start Upload Parameters boundary section
    streamSend.append("--" + boundary + "\r\n");
}
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

```
// Each Parameter is sent within a boundary delimiter
buildUploadParam(streamSend, "message", createXmlMessage(), boundary);

// End form parameters boundary section

// Begin File Upload section

buildUploadFileHeader(streamSend);

// Send Parameters and File header to web server.
postFormFile.write(streamSend.toString().getBytes());

System.out.println("Data Stream Sent\n" + streamSend.toString());
System.out.println("Before data Send");

// Write file to web server
try
{
    writeFileToWeb(postFormFile);
}
catch (IOException e)
{
    System.out.println("Exception in processing input file: " + fileName);
    System.out.println(e);
}

System.out.println("After file Sent");

// Terminate file multipart form-data POST boundary's
streamSend = new StringBuffer();
streamSend.append("\n--" + boundary + "\r\n");
postFormFile.write(streamSend.toString().getBytes());

System.out.println("Data Stream End Sent\n" + streamSend.toString());

// close/cleanup the https output stream
postFormFile.flush();
postFormFile.close();

// Process the web server response.
try
{
    processWebResponse(theURLConnection);
}
catch (IOException e)
{
    System.out.println("Exception in processing web server response ");
    System.out.println(e);
}

System.out.println("HttpMultiPartPost Done.");

}

/**
 * Get the input stream from the Web Server and
 * read the response.  Write Response to standard out.
 *
 * @param theURLConnection URLConnection
 * @exception IOException - thrown for any IO handling errors.
 */
public void processWebResponse(URLConnection theURLConnection)
    throws IOException
{
    // Parse results to ensure file was sent ok."
    // Expecting: "Some sort of html or XML response/confirmation"
    // Read response from post
    // Initialize the input stream to be read from
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

```
InputStream responseFromDestUrl = theURLConnection.getInputStream();

// Build up response into a buffer
StringBuffer thisResponsePage = new StringBuffer();
byte[] respBuffer = new byte[4096];

// The number of bytes read
int bytesRead = 0;

// Iterate to build up response
while ((bytesRead = responseFromDestUrl.read(respBuffer)) >= 0)
{
    thisResponsePage.append(new String(respBuffer).trim());
}

responseFromDestUrl.close(); // Close response stream

// Display the response
System.out.println("Web Server Response: Start");
System.out.println(thisResponsePage.toString());
System.out.println("Web Server Response: End");
}

/**
 * Prompt User for input parameters.
 * This could be done with a Java properties file or
 * an XML config file instead.
 *
 */
public void promptInput()
{
    String currentLine;

    boolean processOptions = true;

    BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));
    try
    {
        System.out.print("User ID:  ");
        System.out.flush();
        currentLine = bufferedReader.readLine();
        if (currentLine != null && !currentLine.equals(""))
        {
            userId = currentLine;
        }

        System.out.println("Password: ");
        System.out.flush();
        currentLine = bufferedReader.readLine();
        if (currentLine != null && !currentLine.equals(""))
        {
            password = currentLine;
        }

        System.out.print("File Path and Name:  ");
        System.out.flush();
        currentLine = bufferedReader.readLine();
        if (currentLine != null && !currentLine.equals(""))
        {
            fileName = currentLine;
            try
            {
                setSize();
            }
            catch (Exception e)
            {
                System.out.println(e);
                System.out.println("aborted");
            }
        }
    }
}
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

```
System.out.print("Proxy Server: (leave blank if none) ");
System.out.flush();
currentLine = bufferedReader.readLine();
if (currentLine != null && !currentLine.equals(""))
{
    proxyServer = currentLine;
    thereIsProxy = true;
}

if (thereIsProxy)
{
    System.out.print("Proxy User Id: ");
    System.out.flush();
    currentLine = bufferedReader.readLine();
    if (currentLine != null && !currentLine.equals(""))
    {
        proxyUser = currentLine;
    }

    System.out.print("Proxy Password: ");
    System.out.flush();
    currentLine = bufferedReader.readLine();
    if (currentLine != null && !currentLine.equals(""))
    {
        proxyPassword = currentLine;
    }
}
catch (IOException e)
{
    System.out.println("IOException " + e);
    System.exit(-1);
}
}

/**
 * Run the process to upload the file. If params are
 * passed from the command then they are processed by
 * acceptArgs, otherwise the user is prompted for input
 * from the promptInput method. Then the postToWeb
 * process is executed.
 *
 * @param args java.lang.String[]
 */
public void runProcess(String[] args)
{
    if (args.length > 0)
    {
        try
        {
            acceptArgs(args);
        }
        catch (Exception e)
        {
            System.out.println(e.getMessage());
            return;
        }
    }
    else
    {
        promptInput();
    }

    try
    {
        postToWeb();
    }
}
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

```
        catch (Exception e)
        {
            System.out.println("Error uploading data, exception is: " + e);
            e.printStackTrace();
        }

    }
/***
 * Sets the file size of the upload file.
 *
 * @exception Exception - thrown if the input XML doesn't exist.
 */
public void setFileSize()
    throws Exception
{
    File checkFile = new File(fileName);

    if (checkFile.exists())
    {
        fileSize = "" + (checkFile.length());
    }
    else
    {
        throw new Exception(fileName + " File does not exist.");
    }
}
/***
 * Sets the HTTP switch to true if the request that is to be processed
 * in non SSL.  SSL requests is the default.
 *
 */
public void setHTTP()
{
    if (System.getProperty("http") == null)
    {
        httpSwitch = false;
    }
    else
    {
        if (System.getProperty("http").toString().equalsIgnoreCase("true"))
        {
            httpSwitch = true;
        }
        else
        {
            httpSwitch = false;
        }
    }
}
/***
 * Sets up the Proxy server properties for the URL Connection.
 * This assumes the proxy server requires authentication.  Not
 * all Proxy servers do, so this will need to be changed.  There
 * is also something called "Digest" which I believe is a different
 * kind of authentication some Proxy servers use that would have to
 * be handled as well.  Port should be a parameter also.
 *
 * @param URLConnection urlConnection
 */
public void setProxy(URLConnection theURLConnection)
{
    if (theURLConnection instanceof com.sun.net.ssl.HttpsURLConnection)
    {
        ((com.sun.net.ssl.HttpsURLConnection) theURLConnection).setSSLSocketFactory(
            new SSLTunnelSocketFactory(proxyServer, proxyPort, proxyUser, proxyPassword));
    }
}
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

```
/**  
 * Reads the file and write's it to the web server.  
 *  
 * @param postFormFile DataOutputStream  
 * @exception IOException - thrown for any IO handling errors.  
 */  
public void writeFileToWeb(DataOutputStream postFormFile)  
    throws java.io.IOException  
{  
    // Read/Write File as bytes  
  
    FileInputStream uploadFileReader = new FileInputStream(fileName);  
    int numBytesToRead = 1024;  
    int availableBytesToRead;  
  
    availableBytesToRead = uploadFileReader.available();  
  
    System.out.println("size of file: " + availableBytesToRead);  
  
    while ((availableBytesToRead = uploadFileReader.available()) > 0)  
    {  
        byte[] bufferBytesRead;  
        // Adjust size of buffered bytes if necessary  
        if (availableBytesToRead >= numBytesToRead)  
        {  
            bufferBytesRead = new byte[numBytesToRead];  
        }  
        else  
        {  
            bufferBytesRead = new byte[availableBytesToRead];  
        }  
  
        // Trap end of file condition  
        int numberOfBytesRead = uploadFileReader.read(bufferBytesRead);  
  
        // Did we reach end of file  
        if (numberOfBytesRead == -1)  
        {  
            break;  
        }  
  
        // Write current buffered bytes to servlet  
        postFormFile.write(bufferBytesRead);  
    } // Iterate through contents of file  
}  
}
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

MassHealth Interface Specification Guide  
August 2005

Version 1.0

## Appendix B – Directory Listing Sample Program

### Directory listing program example

This sample program is presented as a guide only. It is not to be used in production without testing and possible changes. A different version of this sample may be released in response to changes that may occur.



***Important Note: The sample below and other samples in this document are coded in the Java language, but any language that supports the HTTPS protocol can be used to interact with the Web site.***

```
Package com.eds.ne.regional.samples;

import com.sun.net.ssl.HttpsURLConnection;
import com.sun.net.ssl.HostnameVerifier;
import java.net.*;
import java.io.*;

/**
 * This program is provided as sample only as is without warranty or
 * condition of any kind, either expressed or implied, including, but
 * not limited to, the implied warranties of merchantability and
 * fitness for a particular purpose.
 *
 * This sample Directory List program connects to the web server then
 * sends it an XML file containing the directory request message. The
 * name of the file with the XML message is passed to the program,
 * along with the server name. See the acceptArguments method for
 * filenames to pass on the command line. If no parameters are passed
 * the program will prompt the user for the input parameters then
 * execute. The directory listing response is simply displayed in
 * standard out.
 *
 * See the vendor specifications for details of the format of the XML message
 * for a directory listing. Below is a sample of the text that could
 * be in a file passed to this program to request a directory list.
 *
 *
 * <?xml version="1.0" encoding="UTF-8"?>
 * <DirectoryRequest>
 *   <IdentificationHeader>
 *     <SubmitterId>1234567</SubmitterId>
 *     <UserId>1234567</UserId>
 *     <Password>dfdf534</Password>
 *   </IdentificationHeader>
 *   <Transaction>
 *     <Function>DIRLIST</Function>
 *     <FilesToReturnCount>50</FilesToReturnCount>
 *     <SelectedFileTypes>
 *       <FileType>997</FileType>
 *     </SelectedFileTypes>
 *     <FilesCreatedFromDate>20020102</FilesCreatedFromDate>
 *     <FilesCreatedToDate>20020103</FilesCreatedToDate>
 *   </Transaction>
 * </DirectoryRequest>
 *
 *
 * <P>
 * In order to negotiate the SSL certificate of the web server the client has
 * to have a trusted certificate in its Java keystore file. The web sites use
 * GeoTrust Certificates. If this Certificate is not in your Java installations
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

```
* keystore you will need to install it. To install the certificate you can use
* the java keytool command
*
* This sample program was written using JDK 1.2.2. In order for the sample program
* to communicate over SSL the Java Secure Socket Extension (JSSE) needs to be installed.
* See http://java.sun.com/products/jsse/INSTALL.html.
* After installing JSSE add the following jar files must be in the class path before
* running the sample program: drive:/JSSE/jsse.jar;drive:/JSSE/jnet.jar;drive:/JSSE/jcert.jar
* <P>
* <HR>
* <P>
* <TABLE BORDER="1" CELLPADDING="3" CELLSPACING="0" WIDTH="100%">
* <TR BGCOLOR="#9393ff">
*   <TD COLSPAN=4><FONT SIZE="+2"><B>Modification History</B></FONT></TD>
* </TR>
* <TR BGCOLOR="white" CLASS="TableRowColor">
*   <TH>Reason</TH><TH>Date</TH><TH>Systems Engineer</TH><TH>Description of Modification</TH>
* </TR>
* <TR>
*   <TD ALIGN=CENTER>NE Regional HIPAA Translator Implementation</TD>
*   <TD ALIGN=CENTER>12/19/2002</TD>
*   <TD ALIGN=CENTER>EDS</TD>
*   <TD>Initial deployment of this class.</TD>
* </TR>
* </TABLE>
*
*   @author Michael Smith
*
*   @see SSLTunnelSocketFactory
*/
public class ClientDirectoryList
{
/**
 * Authorized User Id. This User Id is authenticated
 * on the server before uploading the users message.
 */
    String userId = "";
/**
 * User Password.
 */
    String password = "";
/**
 * File drive:\path\name.
 */
    String filename = "d:\\temp\\test.xml";
/**
 * File size.
 */
    String fileSize = "0";
/**
 * Users proxy servers name or ip address.
 */
    String proxyServer = null;
/**
 * Users proxys port assignment.
 */
    String proxyPort = "80";
/**
 * True if users sets their proxy server.
 */
    boolean thereIsProxy = false;
/**
 * Users proxy server User Id.
 */
    String proxyUser = null;
/**
 * Users proxy server password.
 */
    String proxyPassword = null;
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

Version 1.0

```
/**  
 * HTTP Protocol Switch used to turn on HTTP only uploads.  
 */  
    boolean httpSwitch = false;  
/**  
 * URL to post the data to. Defaults to test Web Server.  
 */  
    String postUrl = "https:// masshealth2.ehs.state.ma.us  
/transactions/WebDirectoryDownloadFromClient";  
/**  
 * ClientDirectoryList default constructor calls it's super, then  
 * setups System properties for SSL filename.  
 */  
public ClientDirectoryList()  
{  
    super();  
  
    System.setProperty(  
        "java.protocol.handler.pkgs",  
        "com.sun.net.ssl.internal.www.protocol");  
  
    // Required for ssl support and avoiding error:  
    // "unknown protocol: https" exception  
    java.security.Security.addProvider(new com.sun.net.ssl.internal.ssl.Provider());  
}  
/**  
 * Arguments passed on the command line are validated and  
 * set by this method. The parameters are as follows:  
 * <pre>  
 * UserID Password DirectoryListXMLFile [URL] [proxyServer ProxyUser ProxyPassword]  
 * </pre>  
 * Parameters in brackets [] are optional.  
 *  
 * @exception Exception - Invalid arguments message.  
 */  
public void acceptArgs(String[] args)  
    throws Exception  
{  
    if (args.length >= 3)  
    {  
        userId      = args[0];  
        password   = args[1];  
        filename    = args[2];  
    }  
  
    setSize();  
  
    if (args.length == 4)  
    {  
        postUrl      = args[3];  
    }  
  
    if (args.length == 6)  
    {  
        proxyServer   = args[3];  
        proxyUser     = args[4];  
        proxyPassword = args[5];  
        thereIsProxy = true;  
    }  
  
    if (args.length == 7)  
    {  
        postUrl      = args[3];  
        proxyServer   = args[4];  
        proxyUser     = args[5];  
        proxyPassword = args[6];  
        thereIsProxy = true;  
    }  
}
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

```
if (args.length == 3 || args.length == 4 || args.length == 6 || args.length == 7)
{
    // continue
}
else
{
    throw new Exception("Invalid number of parameters found:" + args.length + " must
be 3, 4, 6, or 7 \n" +
                    "Valid Params are: UserID Password FileName [URL] [proxyServer
ProxyUser ProxyPassword]");
}

/**
 * Main creates an instance of ClientDirectoryList then
 * calls its runProcess method.
 *
 * @param args java.lang.String[]
 */
public static void main(String[] args)
{

    ClientDirectoryList clientDirectoryList = new ClientDirectoryList();
    clientDirectoryList.setHTTP();
    clientDirectoryList.runProcess(args);
}

/**
 * Override HostnameVerifier to ignore Certificate errors where the
 * URL on the certificate doesn't match the URL being accessed.
 *
 * @param theURLConnection URLConnection
 */
public void overrideHostNameVerifier(URLConnection theURLConnection)
{
    ((HttpsURLConnection) theURLConnection).setHostnameVerifier(new HostnameVerifier()
    {
        public boolean verify(String urlHost, String certHost)
        {
            if (!urlHost.equals(certHost))
            {
                System.out.println(
                    "certificate <" +
                    certHost +
                    "> does not match host <" +
                    urlHost +
                    "> but " +
                    "continuing anyway");
            }
            return true;
        }
    });
}

/**
 * Post to a web server.
 *
 * @exception MalformedURLException - thrown if URL invalid.
 * @exception IOException - thrown for any IO handling errors.
 */
public void postToWeb() throws MalformedURLException, IOException
{
    // Create URL to post to.
    URL destUrl = new URL(postUrl);

    // Prepare the connection for I the form
    URLConnection theURLConnection = destUrl.openConnection();
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

```
//Setup HTTPS IP Tunneling through Proxy if needed.
If (thereIsProxy)
{
    setProxy(theURLConnection);
}

// Set URL options
theURLConnection.setDoOutput(true);
theURLConnection.setDoInput(true);
theURLConnection.setUseCaches(false);

// Setup Override of error name error if using HTTPS
// and the certificate name doesn't match the URL.
// This code should be removed if the certificate name matches.
If (httpSwitch)
{
    // continue
}
else
{
    // Override Certificate error
    overrideHostNameVerifier(theURLConnection);
}

// Set the XML Post Header
theURLConnection.setRequestProperty(
    "Content-Type",
    "text/xml");

System.out.println("Before DataOutputStream");

DataOutputStream postFormFile =
    new DataOutputStream(theURLConnection.getOutputStream());

System.out.println("Before data Send");

// Write XML file to web server
try
{
    writeXmlRequestToWeb(postFormFile);
}
catch (IOException e)
{
    System.out.println("Exception in processing input file: " + filename);
    System.out.println(e);
}

System.out.println("After file Sent");

// close/cleanup the https output stream
postFormFile.flush();
postFormFile.close();

// Process the web server response.
Try
{
    processWebResponse(theURLConnection);
}
catch (IOException e)
{
    System.out.println("Exception in processing web server response ");
    System.out.println(e);
}

System.out.println("HttpMultiPartPost Done.");

}
/***
 * Get the input stream from the Web Server and
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

```
* read the response.  Write Response to standard out.
/*
 * @param theURLConnection URLConnection
 * @exception IOException - thrown for any IO handling errors.
 */
public void processWebResponse(URLConnection theURLConnection)
    throws IOException
{
    // Parse results to ensure file was sent ok."
    // Expecting: "Some sort of html or XML response/confirmation"
    // Read response from post
    // Initialize the input stream to be read from
    InputStream responseFromDestUrl = theURLConnection.getInputStream();

    // Build up response into a buffer
    StringBuffer thisResponsePage = new StringBuffer();
    byte[] respBuffer = new byte[4096];

    // The number of bytes read
    int bytesRead = 0;

    // Iterate to build up response
    while ((bytesRead = responseFromDestUrl.read(respBuffer)) >= 0)
    {
        thisResponsePage.append(new String(respBuffer).trim());
    }

    responseFromDestUrl.close(); // Close response stream

    // Display the response
    System.out.println("Web Server Response: Start" );
    System.out.println(thisResponsePage.toString());
    System.out.println("Web Server Response: End" );
}
/**/
/* Prompt User for input parameters.
 * This could be done with a Java properties file or
 * an XML config file instead.
 */
public void promptInput()
{
    String currentLine;
    boolean processOptions = true;

    BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));
    try
    {
        System.out.print("User ID:  ");
        System.out.flush();
        currentLine = bufferedReader.readLine();
        if (currentLine != null && !currentLine.equals(""))
        {
            userId = currentLine;
        }

        System.out.println("Password: ");
        System.out.flush();
        currentLine = bufferedReader.readLine();
        if (currentLine != null && !currentLine.equals(""))
        {
            password = currentLine;
        }

        System.out.print("XML File Path and Name:  ");
        System.out.flush();
        currentLine = bufferedReader.readLine();
        if (currentLine != null && !currentLine.equals(""))
        {

```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

```
filename = currentLine;
try
{
    setSize();
}
catch (Exception e)
{
    System.out.println(e);
    System.out.println("aborted");
}

System.out.print("Proxy Server: (leave blank if none) ");
System.out.flush();
currentLine = bufferedReader.readLine();
if (currentLine != null && !currentLine.equals(""))
{
proxyServer = currentLine;
thereIsProxy = true;
}

if (thereIsProxy)
{
    System.out.print("Proxy User Id: ");
    System.out.flush();
    currentLine = bufferedReader.readLine();
    if (currentLine != null && !currentLine.equals(""))
    {
proxyUser = currentLine;
    }

    System.out.print("Proxy Password: ");
    System.out.flush();
    currentLine = bufferedReader.readLine();
    if (currentLine != null && !currentLine.equals(""))
    {
proxyPassword = currentLine;
    }
}
catch (IOException e)
{
    System.out.println("IOException " + e);
    System.exit(-1);
}
}
/** 
 * Run the process to upload the file. If params are
 * passed from the command then they are processed by
 * acceptArgs otherwise the user is prompted for input
 * from the promptInput method. Then the postToWeb
 * process is executed.
 *
 * @param args java.lang.String[]
 */
public void runProcess(String[] args)
{
    if (args.length > 0)
    {
        try
        {
            acceptArgs(args);
        }
        catch (Exception e)
        {
            System.out.println(e.getMessage());
            return;
        }
    }
}
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

```
    }

else
{
    promptInput();
}

try
{
    postToWeb();
}
catch (Exception e)
{
    System.out.println("Error uploading data, exception is: " + e);
    e.printStackTrace();
}

}

/***
 * Sets the file size of the upload file.
 *
 * @exception Exception - thrown if the input XML doesn't exist.
 */
public void setFileSize()
    throws Exception
{
    File checkFile = new File(filename);

    if (checkFile.exists())
    {
        fileSize = "" + (checkFile.length());
    }
    else
    {
        throw new Exception(filename + " File does not exist.");
    }
}
/***
 * Sets the HTTP switch to true if the request that is to be processed
 * is not SSL.  SSL requests is the default.
 */
public void setHTTP()
{
    if (System.getProperty("http") == null)
    {
        httpSwitch = false;
    }
    else
    {
        if (System.getProperty("http").toString().equalsIgnoreCase("true"))
        {
            httpSwitch = true;
        }
        else
        {
            httpSwitch = false;
        }
    }
}
/***
 * Sets up the Proxy server properties for the URL Connection.
 * This assumes the proxy server requires authentication.  Not
 * all Proxy servers do, so this will need to be changed.  There
 * is also something called "Digest" which I believe is a different
 * kind of authentication some Proxy servers use that would have to
 * be handled as well.  The Port number should be a parameter also.
 *
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

```
* @param theURLConnection URLConnection
*/
public void setProxy(URLConnection theURLConnection)
{
    if (theURLConnection instanceof com.sun.net.ssl.HttpsURLConnection)
    {
        ((com.sun.net.ssl.HttpsURLConnection) theURLConnection).setSSLSocketFactory(
            new SSLTunnelSocketFactory(proxyServer, proxyPort, proxyUser, proxyPassword));
    }
}**
 * Writes the XML Request for a directory list to web server.
 *
 * @param postFormFile DataOutputStream
 * @exception IOException - thrown for any IO handling errors.
 */
public void writeXmlRequestToWeb(DataOutputStream postFormFile)
    throws java.io.IOException
{
    // Read/Write File as bytes

    FileInputStream uploadFileReader = new FileInputStream(ilename);
    int numBytesToRead = 1024;
    int availableBytesToRead;

    availableBytesToRead = uploadFileReader.available();

    System.out.println("size of file: " + availableBytesToRead);

    while ((availableBytesToRead = uploadFileReader.available()) > 0)
    {
        byte[] bufferBytesRead;
        // Adjust size of buffered bytes if necessary
        if (availableBytesToRead >= numBytesToRead)
        {
            bufferBytesRead = new byte[numBytesToRead];
        }
        else
        {
            bufferBytesRead = new byte[availableBytesToRead];
        }

        // Trap end of file condition
        int numberOfBytesRead = uploadFileReader.read(bufferBytesRead);

        // Did we reach end of file
        if (numberOfBytesRead == -1)
        {
            break;
        }

        // Write current buffered bytes to servlet
        postFormFile.write(bufferBytesRead);

    } // Iterate through contents of file
}
```

# Commonwealth of Massachusetts Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

Version 1.0

## Appendix C - Download a File Sample Program

### Download a file program example

This sample program is presented as a guide only. It is not to be used in production without testing and possible changes. A different version of this sample may be released in response to changes that may occur.



***Important Note: The sample below and other samples in this document are coded in the Java language, but any language that supports the HTTPS protocol can be used to interact with the Web site.***

```
Package com.eds.ne.regional.samples;

import com.sun.net.ssl.HttpsURLConnection;
import com.sun.net.ssl.HostnameVerifier;
import java.net.*;
import java.io.*;

/**
 * This program is provided as sample only as is without warranty or
 * condition of any kind, either expressed or implied, including, but
 * not limited to, the implied warranties of merchantability and
 * fitness for a particular purpose.
 *
 * This sample File Download program connects to the web server then
 * sends it an XML file containing the download request message. The
 * name of the file with the XML message is passed to the program,
 * along with the server name. See the acceptArguments method for
 * filenames to pass on the command line. If no parameters are passed
 * the program will prompt the user for the input parameters then
 * execute. The downloaded file is simply displayed in standard out.
 *
 * See the vendor specifications for details of the format of the XML message
 * for a file download. Below is a sample of the text that could
 * be in a file passed to this program to request a directory list.
 *
 *
 * <?xml version="1.0" encoding="UTF-8" ?>
 * <DownloadRequest>
 *   <IdentificationHeader>
 *     <SubmitterId>1234567</SubmitterId>
 *     <UserId>1234567</UserId>
 *     <Password>jmstpl#567</Password>
 *   </IdentificationHeader>
 *   <Transaction>
 *     <Function>DOWNLOAD</Function>
 *     <FileName>000000123</FileName>
 *     <FileFormat>TXT</FileFormat>
 *   </Transaction>
 * </DownloadRequest>
 *
 *
 * <P>
 * In order to negotiate the SSL certificate of the web server the client has
 * to have a trusted certificate in its Java keystore file. The web sites use
 * GeoTrust Certificates. If this Certificate is not in your Java installations
 * keystore you will need to install it. To install the certificate you can use
 * the java keytool command.
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

```
*  
* This sample program was written using JDK 1.2.2. In order for the sample program  
* to communicate over SSL the Java Secure Socket Extension (JSSE) needs to be installed.  
* See http://java.sun.com/products/jsse/INSTALL.html.  
* After installing JSSE add the following jar files must be in the class path before  
* running the sample program: drive:/JSSE/jsse.jar;drive:/JSSE/jnet.jar;drive:/JSSE/jcert.jar  
* <P>  
* <HR>  
* <P>  
* <TABLE BORDER="1" CELLPADDING="3" CELLSPACING="0" WIDTH="100%">  
* <TR BGCOLOR="#9393ff">  
* <TD COLSPAN=4><FONT SIZE="+2"><B>Modification History</B></FONT></TD>  
* </TR>  
* <TR BGCOLOR="white" CLASS="TableRowColor">  
* <TH>Reason</TH><TH>Date</TH><TH>Systems Engineer</TH><TH>Description of Modification</TH>  
* </TR>  
* <TR>  
* <TD ALIGN=CENTER>NE Regional HIPAA Translator Implementation</TD>  
* <TD ALIGN=CENTER>12/19/2002</TD>  
* <TD ALIGN=CENTER>EDS</TD>  
* <TD>Initial deployment of this class.</TD>  
* </TR>  
* </TABLE>  
*  
* @author Michael Smith  
*  
* @see SSLSocketFactory  
*/  
public class ClientFileDownload  
{  
/**  
* Users Id. This User Id is authenticated  
* on the server before uploading the users message.  
*/  
    String userId = "";  
/**  
* User password.  
*/  
    String password = "";  
/**  
* Download file name.  
*/  
    String filename = "";  
/**  
* Message file size.  
*/  
    String fileSize = "0";  
/**  
* Users proxy servers name or ip address.  
*/  
    String proxyServer = null;  
/**  
* Users proxys port assignment.  
*/  
    String proxyPort = "80";  
/**  
* True if Users sets their proxy server.  
*/  
    boolean thereIsProxy = false;  
/**  
* Users proxy server User Id.  
*/  
    String proxyUser = null;  
/**  
* Users Proxy Server Password.  
*/  
    String proxyPassword = null;  
/**  
* HTTP filename switch used to turn on HTTP only uploads.  
*/
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

```
/*
 *  @filename httpSwitch = false;
 */
/** 
 * URL to post the data to.
 */
String postUrl = "https://
masshealth2.ehs.state.ma.us/transactions/WebDirectoryDownloadFromClient";
/** 
 * ClientFileDownload default constructor calls it's super, then
 * setups System properties for SSL @filename.
 */
public ClientFileDownload()
{
    super();

    System.setProperty(
        "java.protocol.handler.pkgs",
        "com.sun.net.ssl.internal.www.protocol");

    // Required for ssl support and avoiding error:
    // "unknown protocol: https" exception
    java.security.Security.addProvider(new com.sun.net.ssl.internal.ssl.Provider());
}
/** 
 * Arguments passed on the command line are validated and
 * set by this method.  The parameters are as follows:
 * <pre>
 *   UserID Password FileName [URL] [proxyServer ProxyUser ProxyPassword]
 * </pre>
 *
 * @param args java.lang.String[]
 * @exception Exception - Invalid arguments message.
 */
public void acceptArgs(String[] args)
    throws Exception
{
    if (args.length >= 3)
    {
        userId      = args[0];
        password    = args[1];
        @filename   = args[2];
    }

    setSize();

    if (args.length == 4)
    {
        postUrl      = args[3];
    }

    if (args.length == 6)
    {
        proxyServer   = args[3];
        proxyUser     = args[4];
        proxyPassword = args[5];
        thereIsProxy = true;
    }

    if (args.length == 7)
    {
        postUrl      = args[3];
        proxyServer   = args[4];
        proxyUser     = args[5];
        proxyPassword = args[6];
        thereIsProxy = true;
    }

    if (args.length == 3 || args.length == 4 || args.length == 6 || args.length == 7)
    {

```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

```
// continue
    }
    else
    {
        throw new Exception("Invalid number of parameters found:" + args.length + " must
be 3, 4, 6, or 7 \n" +
                            "Valid Params are: UserID Password FileName [URL] [proxyServer
ProxyUser ProxyPassword]");
    }
}

/**
 * Main creates an instance of ClientFileDownload then
 * calls it's runProcess method.
 *
 * @param args java.lang.String[]
 */
public static void main(String[] args)
{
    ClientFileDownload clientFileDownload = new ClientFileDownload();
    clientFileDownload.setHTTP();
    clientFileDownload.runProcess(args);
}

/**
 * Override Hostname Verifier to ignore Certificate errors where the
 * URL on the certificate doesn't match the URL being accessed.
 *
 * @param theURLConnection URLConnection
 */
public void overrideHostNameVerifier(URLConnection theURLConnection)
{
    ((HttpsURLConnection) theURLConnection).setHostnameVerifier(new HostnameVerifier()
    {
        public boolean verify(String urlHost, String certHost)
        {
            if (!urlHost.equals(certHost))
            {
                System.out.println(
                    "certificate <" +
                    certHost +
                    "> does not match host <" +
                    urlHost +
                    "> but " +
                    "continuing anyway");
            }
            return true;
        }
    });
}

/**
 * Post a request for a file download.
 *
 * @exception MalformedURLException - thrown if URL invalid.
 * @exception IOException - thrown for any IO handling errors.
 */
public void postToWeb() throws MalformedURLException, IOException
{
    // Create URL to post to.
    URL destUrl = new URL(postUrl);

    // Prepare the connection for I the form
    URLConnection theURLConnection = destUrl.openConnection();

    //Setup HTTPS IP Tunneling through Proxy if needed.
    If (thereIsProxy)
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

```
{  
    setProxy(theURLConnection);  
  
}  
  
// Set URL options  
theURLConnection.setDoOutput(true);  
theURLConnection.setDoInput(true);  
theURLConnection.setUseCaches(false);  
  
if (httpSwitch)  
{  
    // continue  
}  
else  
{  
    // Override Certificate error  
    overrideHostNameVerifier(theURLConnection);  
}  
  
// Set the XML Post Header  
theURLConnection.setRequestProperty(  
    "Content-Type",  
    "text/xml");  
  
System.out.println("Before DataOutputStream");  
  
DataOutputStream postFormFile =  
    new DataOutputStream(theURLConnection.getOutputStream());  
  
System.out.println("Before data Send");  
  
// Write file to web server  
try  
{  
    writeXmlRequestToWeb(postFormFile);  
}  
catch (IOException e)  
{  
    System.out.println("Exception in processing input file: " + filename);  
    System.out.println(e);  
}  
  
System.out.println("After file Sent");  
  
// close/cleanup the https output stream  
postFormFile.flush();  
postFormFile.close();  
  
// Process the web server response.  
Try  
{  
    processWebResponse(theURLConnection);  
}  
catch (IOException e)  
{  
    System.out.println("Exception in processing web server response ");  
    System.out.println(e);  
}  
  
System.out.println("postToWeb Done.");  
  
}  
/**  
 * Get the input stream from the Web Server and  
 * read the response. Write Response to standard out.  
 *  
 * @param theURLConnection URLConnection  
 * @exception IOException - thrown for any IO handling errors.
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

```
/*
public void processWebResponse(URLConnection theURLConnection)
    throws IOException
{
    // Parse results to ensure file was sent ok."
    // Expecting: "Some sort of html or XML response/confirmation"
    // Read response from post
    // Initialize the input stream to be read from
    InputStream responseFromDestUrl = theURLConnection.getInputStream();

    // Build up response into a buffer
    StringBuffer thisResponsePage = new StringBuffer();
    byte[] respBuffer = new byte[4096];

    // The number of bytes read
    int bytesRead = 0;

    // Iterate to build up response
    while ((bytesRead = responseFromDestUrl.read(respBuffer)) >= 0)
    {
        thisResponsePage.append(new String(respBuffer).trim());
    }

    responseFromDestUrl.close(); // Close response stream

    // Display the response
    System.out.println("Web Server Response: Start" );
    System.out.println(thisResponsePage.toString());
    System.out.println("Web Server Response: End" );
}
/***
 * Prompt User for input parameters.
 * This could be done with a Java properties file or
 * an XML config file instead.
 */
public void promptInput()
{
    String currentLine;
    boolean processOptions = true;

    BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));
    try
    {
        System.out.print("User ID: ");
        System.out.flush();
        currentLine = bufferedReader.readLine();
        if (currentLine != null && !currentLine.equals(""))
        {
            userId = currentLine;
        }

        System.out.println("Password: ");
        System.out.flush();
        currentLine = bufferedReader.readLine();
        if (currentLine != null && !currentLine.equals(""))
        {
            password = currentLine;
        }

        System.out.print("XML File Path and Name: ");
        System.out.flush();
        currentLine = bufferedReader.readLine();
        if (currentLine != null && !currentLine.equals(""))
        {
            filename = currentLine;
            try
            {

```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

```
        setFileSize();
    }
    catch (Exception e)
    {
        System.out.println(e);
        System.out.println("aborted");
    }
}

System.out.print("Proxy Server: (leave blank if none) ");
System.out.flush();
currentLine = bufferedReader.readLine();
if (currentLine != null && !currentLine.equals(""))
{
proxyServer = currentLine;
thereIsProxy = true;
}

if (thereIsProxy)
{
    System.out.print("Proxy User Id: ");
    System.out.flush();
    currentLine = bufferedReader.readLine();
    if (currentLine != null && !currentLine.equals(""))
    {
proxyUser = currentLine;
    }

    System.out.print("Proxy Password: ");
    System.out.flush();
    currentLine = bufferedReader.readLine();
    if (currentLine != null && !currentLine.equals(""))
    {
proxyPassword = currentLine;
    }
}
catch (IOException e)
{
    System.out.println("IOException " + e);
    System.exit(-1);
}
}
/** 
 * Run the process to upload the file.  If params are
 * passed from the command then they are processed by
 * acceptArgs otherwise the user is prompted for input
 * from the promptInput method.  Then the postToWeb
 * process is executed.
 *
 * @param args java.lang.String[]
 */
public void runProcess(String[] args)
{
    if (args.length > 0)
    {
        try
        {
            acceptArgs(args);
        }
        catch (Exception e)
        {
            System.out.println(e.getMessage());
            return;
        }
    }
    else
    {
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

```
        promptInput();
    }

    try
    {
        postToWeb();
    }
    catch (Exception e)
    {
        System.out.println("Error uploading data, exception is: " + e);
        e.printStackTrace();
    }

}

/**
 * Sets the file size of the upload file.
 *
 * @exception Exception - thrown if the input XML doesn't exist.
 */
public void setFileSize()
    throws Exception
{
    File checkFile = new File(filename);

    if (checkFile.exists())
    {
        fileSize = "" + (checkFile.length());
    }
    else
    {
        throw new Exception(filename + " File does not exist.");
    }
}

/**
 * Sets the HTTP switch to true if the request that is to be processed
 * in non SSL.  SSL requests is the default.
 *
 */
public void setHTTP()
{
    if (System.getProperty("http") == null)
    {
        httpSwitch = false;
    }
    else
    {
        if (System.getProperty("http").toString().equalsIgnoreCase("true"))
        {
            httpSwitch = true;
        }
        else
        {
            httpSwitch = false;
        }
    }
}

/**
 * Sets up the Proxy server properties for the URL Connection.
 * This assumes the proxy server requires authentication.  Not
 * all Proxy servers do, so this will need to be changed.  There
 * is also something called "Digest" which I believe is a different
 * kind of authentication some Proxy servers use that would have to
 * be handled as well.  Port should be a parameter also.
 *
 * @param theURLConnection URLConnection
 */
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

```
public void setProxy(URLConnection theURLConnection)
{
    if (theURLConnection instanceof com.sun.net.ssl.HttpsURLConnection)
    {
        ((com.sun.net.ssl.HttpsURLConnection) theURLConnection).setSSLSocketFactory(
            new SSLTunnelSocketFactory(proxyServer, proxyPort, proxyUser, proxyPassword));
    }
}**
 * Writes the XML Request for a file to web server.
 *
 * @param postFormFile DataOutputStream
 * @exception java.io.IOException The exception description.
 */
public void writeXmlRequestToWeb(DataOutputStream postFormFile)
    throws java.io.IOException
{
    // Read/Write File as bytes

    FileInputStream uploadFileReader = new FileInputStream(filename);
    int numBytesToRead = 1024;
    int availableBytesToRead;

    availableBytesToRead = uploadFileReader.available();

    System.out.println("size of file: " + availableBytesToRead);

    while ((availableBytesToRead = uploadFileReader.available()) > 0)
    {
        byte[] bufferBytesRead;
        // Adjust size of buffered bytes if necessary
        if (availableBytesToRead >= numBytesToRead)
        {
            bufferBytesRead = new byte[numBytesToRead];
        }
        else
        {
            bufferBytesRead = new byte[availableBytesToRead];
        }

        // Trap end of file condition
        int numberofBytesRead = uploadFileReader.read(bufferBytesRead);

        // Did we reach end of file
        if (numberofBytesRead == -1)
        {
            break;
        }

        // Write current buffered bytes to servlet
        postFormFile.write(bufferBytesRead);

    } // Iterate through contents of file
}
```

# Commonwealth of Massachusetts Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

Version 1.0

## Appendix D - SSL Tunnel Through Proxy Server Utility Sample

### Program Sample

This sample program is **needed only if** you have to connect through a proxy server to get to the Internet.

This sample program is presented as a guide only. It is not to be used in production without testing and possible changes. A different version of this sample may be released in response to changes that may occur.



***Important Note: The sample below and other samples in this document are coded in the Java language, but any language that supports the HTTPS protocol can be used to interact with the Web site.***

```
Package com.eds.ne.regional.samples;

import java.net.*;
import java.io.*;
import java.security.*;
import sun.misc.BASE64Encoder;
import javax.net.*;
import javax.net.ssl.*;

/**
 * This program is provided as sample only as is without warranty or
 * condition of any kind, either expressed or implied, including, but
 * not limited to, the implied warranties of merchantability and
 * fitness for a particular purpose.
 *
 * This sample program is used to facilitate HTTPS (SSL) communication when
 * connecting through a Proxy Server.
 *
 *      <P>
 *      <HR>
 *      <P>
 *      <TABLE BORDER="1" CELLPADDING="3" CELLSPACING="0" WIDTH="100%">
 *      <TR BGCOLOR="#9393ff">
 *          <TD COLSPAN=4><FONT SIZE="+2"><B>Modification History</B></FONT></TD>
 *      </TR>
 *      <TR BGCOLOR="white" CLASS="TableRowColor">
 *          <TH>Reason</TH><TH>Date</TH><TH>Systems Engineer</TH><TH>Description of Modification</TH>
 *      </TR>
 *      <TR>
 *          <TD ALIGN=LEFT>NE Regional HIPAA Translator Implementation</TD>
 *          <TD ALIGN=LEFT>12/19/2002</TD>
 *          <TD ALIGN=LEFT>EDS</TD>
 *          <TD>Initial deployment of this class.</TD>
 *      </TR>
 *      </TABLE>
 *
 *      @see SSLSocketFactory
 */
public class SSLSocketFactory extends SSLSocketFactory
{
    /**
     * Proxy server Host name.
     */
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

Version 1.0

```
private String tunnelHost;
    /**
     * Proxy server port assignment.
     */
private int tunnelPort;
    /**
     * SSL Socket Factory returns a default Socket configured for SSL communication.
     */
private SSLSocketFactory dfactory;
    /**
     * Proxy Server password used in tunneling.
     */
private String tunnelPassword;
    /**
     * Proxy Server User Id used in tunneling.
     */
private String tunnelUserName;
    /**
     * Switch to indicate we have a connection.
     */
private boolean socketConnected = false;

/**
 * Constructor for the SSLTunnelSocketFactory object.
 *
 * @param proxyHost The url of the proxy host
 * @param proxyPort the port of the proxy
 */
public SSLTunnelSocketFactory(String proxyHost, String proxyPort)
{
    System.out.println("SSLTunnelSocketFactory: Creating Socket Factory");

    tunnelHost = proxyHost;
    tunnelPort = Integer.parseInt(proxyPort);

    dfactory = (SSLSocketFactory) SSLSocketFactory.getDefault();

}
/**
 * Default constructor for the SSLTunnelSocketFactory object.
 *
 * @param proxyHost      The url of the proxy host
 * @param proxyPort      the port of the proxy
 * @param proxyUserName  username for authenticating with the proxy
 * @param proxyPassword  password for authenticating with the proxy
 */
public SSLTunnelSocketFactory(
    String proxyHost,
    String proxyPort,
    String proxyUserName,
    String proxyPassword)
{
    System.out.println("creating Socket Factory with password/username");
    tunnelHost = proxyHost;
    tunnelPort = Integer.parseInt(proxyPort);
    tunnelUserName = proxyUserName;
    tunnelPassword = proxyPassword;
    dfactory = (SSLSocketFactory) SSLSocketFactory.getDefault();
}
/**
 * Creates a new SSL Tunneled Socket
 *
 * @param host            destination host
 * @param port            destination port
 * @return                tunneled SSL Socket
 * @exception IOException raised by IO error
 * @exception UnknownHostException raised when the host is unknown
 */
public Socket createSocket(String host, int port)
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

```
    throws IOException, UnknownHostException
{
    return createSocket(null, host, port, true);
}
/**
 * Creates a new SSL Tunneled Socket
 *
 * @param host           Destination Host
 * @param port           Destination Port
 * @param clientHost    Ignored
 * @param clientPort    Ignored
 * @return              SSL Tunneled Socket
 * @exception IOException Raised when IO error occurs
 * @exception UnknownHostException Raised when the destination host is unknown
 */
public Socket createSocket(
    String host,
    int port,
    InetAddress clientHost,
    int clientPort)
    throws IOException, UnknownHostException
{
    return createSocket(null, host, port, true);
}
/**
 * Creates a new SSL Tunneled Socket.
 *
 * @param host   destination host
 * @param port   destination port
 * @return      tunneled SSL Socket
 * @exception IOException raised when IO error occurs
 */
public Socket createSocket(InetAddress host, int port)
    throws IOException
{
    return createSocket(null, host.getHostName(), port, true);
}
/**
 * Creates a new SSL Tunneled Socket
 *
 * @param address      destination host
 * @param port         destination port
 * @param clientAddress ignored
 * @param clientPort   ignored
 * @return              tunneled SSL Socket
 * @exception IOException raised when IO exception occurs
 */
public Socket createSocket(
    InetAddress address,
    int port,
    InetAddress clientAddress,
    int clientPort)
    throws IOException
{
    return createSocket(null, address.getHostName(), port, true);
}
/**
 * Creates a new SSL Tunneled Socket
 *
 * @param s           Ignored
 * @param host        destination host
 * @param port        destination port
 * @param autoClose   wether to close the socket automatically
 * @return             proxy tunneled socket
 * @exception IOException raised by an IO error
 * @exception UnknownHostException raised when the host is unknown
 */
public Socket createSocket(Socket s, String host, int port, boolean autoClose)
    throws IOException, UnknownHostException
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

MassHealth Interface Specification Guide  
August 2005

Version 1.0

```
{  
    Socket tunnel = new Socket(tunnelHost, tunnelPort);  
    doTunnelHandshake(tunnel, host, port);  
  
    SSLSocket result =  
        (SSLSocket) dfactory.createSocket(tunnel, host, port, autoClose);  
  
    result.addHandshakeCompletedListener(new HandshakeCompletedListener()  
    {  
        public void handshakeCompleted(HandshakeCompletedEvent event)  
        {  
            System.out.println("Handshake Finished!");  
            System.out.println("\t CipherSuite :" + event.getCipherSuite());  
            System.out.println("\t SessionId: " + event.getSession());  
            System.out.println("\t PeerHost: " + event.getSession().getPeerHost());  
            setSocketConnected(true);  
        }  
    });  
  
    // thanks to David Lord in the java forums for figuring out this line is the problem  
    // result.startHandshake(); //this line is the bug which stops Tip111 from working correctly  
  
    return result;  
}  
/**  
 * Description of the Method  
 *  
 * @param tunnel          tunnel socket  
 * @param host            destination host  
 * @param port            destination port  
 * @exception IOException raised when an IO error occurs  
 */  
private void doTunnelHandshake(Socket tunnel, String host, int port)  
    throws IOException  
{  
    OutputStream out = tunnel.getOutputStream();  
  
    //generate connection string  
    String msg =  
        "CONNECT "  
        + host  
        + ":"  
        + port  
        + " HTTP/1.0\r\n"  
        + "User-Agent: "  
        + sun.net.www.protocol.http.HttpURLConnection.userAgent;  
  
    if (tunnelUserName != null && tunnelPassword != null)  
    {  
        //add basic authentication header for the proxy  
        sun.misc.BASE64Encoder enc = new sun.misc.BASE64Encoder();  
        String encodedPassword = enc.encode((tunnelUserName +  
                                              ":" + tunnelPassword).getBytes());  
        msg = msg + "\nProxy-Authorization: Basic " + encodedPassword;  
    }  
  
    msg = msg + "\nContent-Length: 0";  
    msg = msg + "\nPragma: no-cache";  
    msg = msg + "\r\n\r\n";  
    System.out.println("doTunnelHandshake Message:" + msg);  
  
    byte b[];  
  
    try  
    {  
        //we really do want ASCII7 as the http protocol doesnt change with locale  
        b = msg.getBytes("ASCII7");  
    }  
    catch (UnsupportedEncodingException ignored)
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

Version 1.0

```
{  
    //If ASCII7 isn't there, something is seriously wrong!  
    b = msg.getBytes();  
}  
  
out.write(b);  
out.flush();  
  
byte reply[] = new byte[200];  
int replyLen = 0;  
int newlinesSeen = 0;  
boolean headerDone = false;  
  
InputStream in = tunnel.getInputStream();  
  
boolean error = false;  
  
while (newlinesSeen < 2)  
{  
    int i = in.read();  
    if (i < 0)  
    {  
        throw new IOException("Unexpected EOF from Proxy");  
    }  
  
    if (i == '\n')  
    {  
        headerDone = true;  
        ++newlinesSeen;  
    }  
    else  
    {  
        if (i != '\r')  
        {  
            newlinesSeen = 0;  
            if (!headerDone && replyLen < reply.length)  
            {  
                reply[replyLen++] = (byte) i;  
            }  
        }  
    }  
}  
  
//convert byte array to string  
String replyStr;  
  
try  
{  
    replyStr = new String(reply, 0, replyLen, "ASCII7");  
}  
catch (UnsupportedEncodingException ignored)  
{  
    replyStr = new String(reply, 0, replyLen);  
}  
  
//we check for connection established because our proxy returns http/1.1 instead of 1.0  
if (replyStr.toLowerCase().indexOf("200 connection established") == -1)  
{  
    System.out.println("doTunnelHandshake:" + replyStr);  
    throw new IOException(  
        "Unable to tunnel through "  
        + tunnelHost  
        + ":"  
        + tunnelPort  
        + ". Proxy returns\""  
        + replyStr  
        + "\"");  
}  
//tunneling handshake was successful
```

# Commonwealth of Massachusetts

## Executive Office of Health and Human Services

*MassHealth Interface Specification Guide*  
August 2005

*Version 1.0*

```
}

/**
 * Gets the defaultCipherSuites attribute of the SSLTunnelSocketFactory
 * object.
 *
 * @return      The defaultCipherSuites value
 */
public String[] getDefaultCipherSuites()
{
    return dfactory.getDefaultCipherSuites();
}
/**
 * Gets the socketConnected attribute of the SSLTunnelSocketFactory object
 *
 * @return      The socketConnected value
 */
public synchronized boolean getSocketConnected()
{
    return socketConnected;
}
/**
 * Gets the supportedCipherSuites attribute of the SSLTunnelSocketFactory
 * object.
 *
 * @return      The supportedCipherSuites value
 */
public String[] getSupportedCipherSuites()
{
    return dfactory.getSupportedCipherSuites();
}
/**
 * Sets the proxyPassword attribute of the SSLTunnelSocketFactory object
 *
 * @param proxyPassword  The new proxyPassword value
 */
public void setProxyPassword(String proxyPassword)
{
    tunnelPassword = proxyPassword;
}
/**
 * Sets the proxyUserName attribute of the SSLTunnelSocketFactory object
 *
 * @param proxyUserName  The new proxyUserName value
 */
public void setProxyUserName(String proxyUserName)
{
    tunnelUserName = proxyUserName;
}
/**
 * Sets the socketConnected attribute of the SSLTunnelSocketFactory object
 *
 * @param b  The new socketConnected value
 */
private synchronized void setSocketConnected(boolean b)
{
    socketConnected = b;
}
```